DOCUMENT RESUME

ED 425 731 IR 019 219

AUTHOR Snyder, Robin M.
TITLE Using Frames and JavaScript To Automate Teacher-Side Web Page Navigation for Classroom Presentations.
PUB DATE 1998-00-00
NOTE 10p.; In: Association of Small Computer Users in Education: Proceedings of the ASCUE Summer Conference (31st, North Myrtle Beach, SC, June 7-11, 1998); see IR 019 201.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Access to Information; Automation; Computer Assisted Instruction; Computer Oriented Programs; Higher Education; *Hypermedia; Information Retrieval; *Multimedia Instruction; *Multimedia Materials; *Navigation (Information Systems); Programming; Programming Languages; Screen Design (Computers); *World Wide Web
IDENTIFIERS Browsing; *HTML; *Java Programming Language; Web Pages

ABSTRACT

HTML provides a platform-independent way of creating and making multimedia presentations for classroom instruction and making that content available on the Internet. However, time in class is very valuable, so that any way to automate or otherwise assist the presenter in Web page navigation during class can save valuable seconds. This paper describes the use of frames and JavaScript in conjunction with VGA/TV project systems to automate teacher-side Web page navigation for classroom presentations. The methods used are not visually distracting to the students during class, do not require Web server support or an Internet connection (i.e., they can be used with only local hard drive access), and can be used independently of the student-side Web page access so that students who may or may not have browsers that support the features used on the teacher side can still access the content of those pages. (Author/AEF)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

Using frames and JavaScript to automate teacher-side web page navigation for classroom presentations

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
C.P. Singer
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Robin M. Snyder
Byrd School of Business
Shenandoah University
Winchester, VA 22601
rsnyder@su.edu

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED 425 731

ABSTRACT

HTML provides a platform-independent way of creating and making multimedia presentations for classroom instruction and making that content available on the Internet. However, time in class is very valuable, so that any way to automate or otherwise assist the presenter in web page navigation during class can save valuable seconds. This paper (and presentation) will describe (and demonstrate) the use of frames and JavaScript in conjunction with VGA/TV project systems to automate teacher-side web page navigation for classroom presentations. The methods used are not visually distracting to the students during class, do not require web server support or an Internet connection (i.e., it can be used with only local hard drive access), and can be used independently of the student-side web page access so that students who may or may not have browsers that support the features used on the teacher side can still access the content of those pages.

THE WEB SYSTEM

We will assume that the the teacher has a way of creating presentations or other web-based material in HTML format that is to be presented in class using a web browser such that the students can see the presentation. There are many ways to display the screen such that it can be seen by students. The author uses the low-cost way of using one or more 27-inch TVs that display the screen output of a laptop computer via a SVGA-to-TV converter.

The entire web system is designed such that all files in the web system, except for an `index.htm` file in the root that requires special handling, are in directories that are immediately below the root. Thus, there is only one level of subdirectories, no other subdirectories. Thus, any HTML file can reference any other file within the web system by first going up to the parent directory (i.e., the root) via "..", then down to the desired directory. For technical reasons related to insuring web system integrity, this is done even if the referenced file is in the same directory as the referencing file. This design principle makes the entire web system simple and entirely relocatable. As there are no absolute file references, the web system on the local hard drive is the same as the web system on the network is the same as the web system as seen from the Internet. This provides convenience in developing and updating the web systems. For example, even if the Internet and the network go down, the class can proceed by using the copy on the local hard drive. In addition, throughput is improved since access to the local hard drive is always faster than access to files on the network which is always faster than access to files on the Internet.

FRAMES AND SCREENS

A frame provides a way to divide a browser window into several panes. Thus, more than one web

page can be displayed at the same time. More importantly, events in one frame can cause other frames to be reloaded with new pages.

There are many way to arrange frames. The author uses a SVGA-to- TV converter that takes the 800 by 600 SVGA output from the laptop computer and converts it for TV output, which can handle a size of about 600 by 440, slightly smaller than a standard 640 by 480 VGA display. Unless the entire 800 by 600 screen is to be displayed to the students, making most of the text on the screen hard to read, the 600 by 440 area in the upper right of the 800 by 600 screen is displayed to the students. This leaves a margin of 200 pixels on the left and 160 pixels on the bottom that can be used for special navigation features.

The Windows 95 taskbar provides a move convenient way to switch between tasks in a classroom setting over using the `Alt-Tab` feature which can be visually distracting. This leaves about 110 pixels on the bottom part of the screen for navigation controls.

Application programs such as Excel and Word can be customized such that toolbars and buttons are arranged in this navigation area. But, the topic of this paper is how frames and JavaScript can be used to provide automate teacher-side web page navigation for classroom presentations.

The following is a simple HTML file.

```
<HTML>
<HEAD>
<TITLE>Title bar caption goes here</TITLE>
</HEAD>
<BODY>
Text, images, etc., to be browsed goes here.
</BODY>
</HTML>
```

The following is the frame file used by the author. This file is specified as the home page file in NetScape and Internet Explorer. Note that there is no `BODY` section when using frames.

```
<HTML>
<HEAD>
<TITLE>Kithara System, by Robin Snyder</TITLE>
</HEAD>
<FRAMESET COLS="15%,85%">
   <FRAME NAME="LEFTFRAME"
      SRC="../RMS1.WEB/left.htm"
      SCROLLING="no" MARGINWIDTH=0>
   <FRAMESET ROWS="80%,20%">
      <FRAME NAME="MAINFRAME"
      SRC="../RMS.WEB/index.htm">
      <FRAME NAME="BOTFRAME"
      SRC="../RMS1.WEB/bottom.htm"
      SCROLLING="no" MARGINWIDTH=0>
   </FRAMESET>
</FRAMESET>
</HTML>
```

This setup allocates the frame called LEFTFRAME to the leftmost 15% of the screen area, and the remaining 75% is allocated to the frameset that consists of the frame called MAINFRAME which has the upper 80% of that area and the frame called BOTFRAME that has the lower 20% of that area.

Note that the title

```
Kithara System, by Robin Snyder
```

is the title that appears on the title bar. Such a title can be useful in locating, via program code, the appropriate window to switch to if such switching is to be automated.

The MAINFRAME is used to display the presentation to the students. Command buttons that use JavaScript to provide navigation among the author's web system are placed into the LEFTFRAME and BOTFRAME frames. Initially, LEFTFRAME will display the HTML file `left.htm`, BOTFRAME will display the HTML file `bottom.htm`, while MAINFRAME will display `index.htm`, the author's home page.

To conserve precious screen space, neither LEFTFRAME or BOTFRAME have SCROLLING set to NO (i.e., no scrollbars) and the MARGINWIDTH is set to 0. The teacher needs to insure that it is not necessary to scroll these frames. This brings up an important design consideration. While the pages displayed in MAINFRAME will be accessed by students and should therefore contain portable HTML formatting code, everything in LEFTFRAME or BOTFRAME will be used only by the teacher. Therefore, these frames need not be portable but need only work on the teacher's computer. Thus, if the JavaScript code used works only with NetScape and not with Internet Explorer, this is not a problem since the teacher will be using only NetScape for presentations and not Internet Explorer.

The TARGET feature of HTML links can be used to change a page in a frame. However, as we will see, it turns out to be more generally useful to use input buttons and JavaScript with NetScape, since functionality in addition to just loading a page can be added. Similar things could be done using VBScript with Internet Explorer, but that is beyond the scope of this paper.

HTML FORMS

Input buttons and text boxes must be placed on a form. Since these are only for navigation purposes, we do not need a SUBMIT button. The FORM tag has the following form and is placed within the BODY section of the HTML file.

```
<FORM
   NAME="FORM1"
   ACTION="mailto:rsnyder@su.edu"
   METHOD=POST
   >

</FORM>
```

Since we are not using the ACTION or METHOD attributes, the preceding can be simplified to the following.

```
<FORM NAME="identifier">

</FORM>
```

The NAME attribute specifies the "**identifier**" by which JavaScript refers to the form. The author uses the name FORM1, since this is the only form of concern here.

The INPUT tag is used to place an input button on the form and has the following form.

```
<INPUT
   TYPE="BUTTON"
   VALUE="caption"
   NAME="identifier"
   ONCLICK="function"
   >
```

The NAME attribute specifies the "**identifier**" by which JavaScript refers to the button. The VALUE attribute specifies the "**caption**" text which is displayed on the button. The ONCLICK attribute specifies the JavaScript "**function**", with parameters, that is called when the button is clicked.

The INPUT tag is also used to place a text box on the form and has the following form.

```
<INPUT
   TYPE="TEXT"
   NAME="identifier"
   SIZE=36
   >
```

The NAME attribute specifies the "**identifier**" by which JavaScript refers to the text box. The SIZE attribute specifies the width of the text box.

SCRIPTS

JavaScript is a programming language whose commands work within a browser to perform tasks involving HTML web pages and the Internet. For security reasons, both JavaScript is extremely limited as to what can be written to the the local hard drive.

The SCRIPT tag is used to specify the use of JavaScript.

```
<SCRIPT LANGUAGE="JAVASCRIPT"
   SRC="../RMS1.WEB/tswitch.js">
</SCRIPT>
```

Since the navigation capabilities to be added are being used by the teacher and not the students, it is only necessary that the LEFT and BOTTOM frames work with whatever browser the teacher is using. Since both of these frames will use the same JavaScript run-time system, the SCR attribute

is used to refer to the JavaScript text file `tswitch.js` that contains the JavaScript code. Since the code consists only of function definitions and, thus, need not be executed as the page is loading, the `SCRIPT` tag can be placed either in the `HEAD` or `BODY` of the HTML file.

JAVASCRIPT

The functions that comprise the JavaScript runtime system are now presented, along with an example, where appropriate, of the function call that uses that function from the HTML page.

Whenever a page is loaded into another pane of a frame, the user must click on that pane to give it the focus so that the page up/down keys can be used for navigation purposes. The `set_focus` function is used, where deemed appropriate, to set the focus to the `MAINFRAME` pane, saving a mouse click on that pane whenever a button is clicked.

```
function set_focus() {
   top.MAINFRAME.focus()
   }
```

WARNING: JavaScript is case sensitive, so `focus` is not the same as `Focus`. Beware.

Notice that `top` is the top-level page, `MAINFRAME` is one of the panes in the frame, and `focus` is a method of the frame pane.

A common use of a button is to go to a specified page. The function `goto_page` is used to go to page `s`.

```
function goto_page(s) {
   top.MAINFRAME.document.location = s
   set_focus()
   }
```

The `document.location` of `MAINFRAME` is set to `s`, loading that page, and then the `MAINFRAME` is given the focus. For security reasons, this loading only works from the local hard drive, but that is the intended use of the function.

An example of an `INPUT` button tag is as follows.

```
<INPUT
   TYPE="button"
   VALUE="Home"
   NAME="Home"
   ONCLICK="goto_page('../RMS.WEB/index.htm')"
   >
```

When clicked, this button will go to the author's local home page at `../RMS.WEB/index.htm` and set the focus to the `MAINFRAME` pane.

When using frames, it can be difficult to determine what file is loaded into the `MAINFRAME`. A text

box is used to MARK the currently loaded page, displaying the URL in a text box. A GO button provides a way that the teacher can use to return to this page. The HTML tags to create a MARK button, a text box, and a GO button are as follows.

```
<INPUT
   TYPE="button"
   VALUE="Mark"
   NAME="MARK1"
   ONCLICK="mark_url1()">

<INPUT
   TYPE="TEXT"
   NAME="TEXT1"
   SIZE=36>

<INPUT
   TYPE="button"
   VALUE="Go"
   NAME="GOTO1"
   ONCLICK="goto_url1()"
   >
```

The author actually uses three sets of these, but only one is covered here. For security reasons, one can only MARK the URL of a file from the local hard drive.

The JavaScript code for mark_url1 is as follows.

```
function mark_url(t) {
   t.value = top.MAINFRAME.document.location.href
   set_focus()
   }

function mark_url1() {
   mark_url(top.BOTFRAME.document.FORM1.TEXT1)
   }
```

Since the text box requires a lot of horizontal space, it is placed in the BOTFRAME and not the LEFTFRAME. When the MARK button is pressed, the caption of the text box TEXT1 is set to the current URL. The function mark_url1 is used to make it easy to add additional capability.

The JavaScript code for goto_url1 is as follows.

```
function goto_url(fs) {
   if (fs == "") {
      set_focus()
      }
   else {
      goto_page(fs)
      }
   }

function goto_url1() {
   goto_url(top.BOTFRAME.document.FORM1.TEXT1.value)
}
```

If the text box is empty, then the focus is set to the MAINFRAME, and no page is loaded. Note that the function `goto_url` is used to make it easy to add additional capability.

The author's word processor generates HTML. A word processor macro was written to create the `$goto$.htm` file that loads the just-generated HTML file into the browser and then switch to the browser. That is, clicking on the `goto` button calls the `goto_link` function that loads the `$goto$.htm` file into the MAINFRAME pane. The `goto_link` function is used to go to the page `$goto$.htm` on the local hard drive and appears as follows.

```
function goto_link() {
  top.MAINFRAME.location.replace('file:///D|/RMS1.WEB/$goto$.htm')
  set_focus()
        }
```

An example of an INPUT button tag is as follows.

```
<INPUT
   TYPE="button"
   VALUE="goto"
   NAME="goto"
   ONCLICK="goto_link()"
   >
```

An example of the file `$goto$.htm` is as follows.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JAVASCRIPT">
top.MAINFRAME.location.replace('file:///D|/RMS1.WEB/index.ht
m#3h')
top.MAINFRAME.focus()
</SCRIPT>
</BODY>
</HTML>
```

Note that, in this case, the JavaScript code appears inline and is not included as a separate file. In addition, a target is added to the file by the word processor macro, in this case "#3h", so that the location in the web page just loaded is approximately the corresponding place in the document that was used to generate the HTML. This method makes it easy to generate HTML and then switch to the browser to see what the HTML looks like.

Due to security restrictions, however, it is not easy to go from an HTML page in the browser to the word processing document that was used to create that file. The best the author has done is to create a button that fills a text box with the URL of the page. This text box can then be copied to the clipboard, the word processor can be made active (e.g., via `Alt-Tab`), and a macro used to use the

contents of the clipboard to determine and load the appropriate document. This is cumbersome, but when dealing with over a thousand document files, it is better than manually searching for the desired file.

The author's word processor is used to generate both an HTML file that can be used for classroom presentation (i.e., it is slide- based with larger-sized fonts and contains questions and answers to be used in class) or to an HTML file for student use (i.e., it uses smaller fonts and does not contain the answers to questions asked in class). For example, if `F:\IS207\java-08.spr` is the document file, then the HTML presentation file would be `D:\IS207.HTM\java-08.htm` and the student accessible HTML file would be `D:\IS207.WEB\java-08.htm`. The presentation files (with selected answers to questions) remain on the author's local hard drive while the student accessible files are automatically updated to the network and, therefore, the Internet.

The following is a more involved function called `talk_switch` that can be used to automatically switch between the presentation version and the student accessible version of the generated HTML files.

```
function talk_switch() {

   var d = top.MAINFRAME.document
   var fs = d.location.href

   var i = fs.indexOf('.WEB/')
   var j = fs.indexOf('.HTM/')
   var k = fs.length

   if (i != -1) {
      fs = fs.substring(0,i) + ".HTM/" + fs.substring(i+5,k)
      d.location = fs
      }

   if (j != -1) {
      fs = fs.substring(0,j) + ".WEB/" + fs.substring(j+5,k)
      d.location = fs
      }

   set_focus()

   mark_url2()
   }
```

WARNING: Again, JavaScript is case sensitive, so `indexof` is not the same as `indexOf` is not the same thing as `IndexOf`. Beware!

Briefly, the string variable `fs` contains the file. If the directory extension is `.WEB/`, it is changed to `.HTM/`. If the directory extension is `.HTM/`, it is changed to `.WEB/`. Then that other file is loaded and the focus is set.

The `INPUT` tag is as follows.

```
<INPUT
   TYPE="button"
   VALUE="Talk"
   NAME="Talk"
   ONCLICK="talk_switch()"
   >
```

CONCLUSIONS

This paper has covered just a few of the possible uses of JavaScript as a way to automate teacher-side web page navigation for classroom presentations. Future enhancements include investigating the use of Java for providing useful capabilities and creating a search mechanism for quickly locating topics in the author's local web system. The JavaScript code presented has worked well in practice, but there are surely better ways to write the code. These and other methods not covered here have improved and continue to improve the bringing of technology into the classroom. The only limitation is your imagination and the security restrictions imposed by the browser.